VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

NÁHODNÝ GENERÁTOR ČÍSEL NA BÁZI ŠUMU PRO NÍZKOVÝKONNOVÉ ZAŘÍZENÍ

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

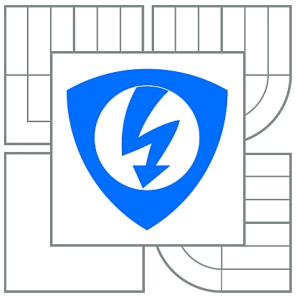
AUTOR PRÁCE
AUTHOR

TOMÁŠ ECLER

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

NÁHODNÝ GENERÁTOR ČÍSEL NA BÁZI ŠUMU PRO NÍZKOVÝKONNOVÉ ZAŘÍZENÍ

RANDOM NUMBER GENERATOR BASED ON NOISE FOR LOW-POWER DEVICES

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

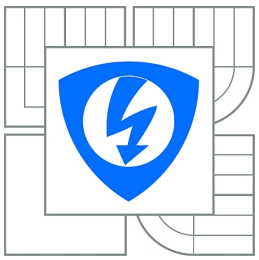
AUTOR PRÁCE — TOMÁŠ ECLER

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR MLÝNEK, Ph.D.

SUPERVISOR

BRNO 2015



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Tomáš Ecler	***ID:***	155155
Ročník:	3	***Akademický rok:***	2014/2015

NÁZEV TÉMATU:

Náhodný generátor čísel na bázi šumu pro nízkovýkonnové zařízení

POKYNY PRO VYPRACOVÁNÍ:

Student v bakalářské práci bude mít za úkol vytvořit hardwarový generátor náhodných čísel pro MSP430. Generátor může být založen například na generování skrze frekvenční šum. Dále bude mít student za úkol ověřit tento generátor jedním z volně dostupných standardizovaných prostředků pro ověřování generátorů náhodných čísel a také proměřit a popř. optimalizovat rychlost generování čísel.

DOPORUČENÁ LITERATURA:

[1] BURDA, K. Bezpečnost informačních systémů. Brno: VUT v Brně, 2013. s. 1-152. ISBN: 978-80-214-4890- 2.

Termín zadání:	9.2.2015	***Termín odevzdání:***	2.6.2015

Vedoucí práce: Ing. Petr Mlýnek, Ph.D.
Konzultanti bakalářské práce:

doc. Ing. Jiří Mišurec, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Bakalářská práce se zabývá vlastním návrhem hardwarového generátoru náhodných čísel s využitím nízko-výkonového mikrokontroleru MSP430. Nejprve jsou popsány stávající způsoby generování náhodných čísel a srovnání různých typů generátorů. Následně jsou uvedeny baterie statistických testů, kterými lze ověřit generátor na základě jeho vygenerované posloupnosti. Závěrem práce je pak ověření vlastního návrhu generátoru statistickými testy NIST.

KLÍČOVÁ SLOVA

Skutečný generátor náhodných čísel, generátor pseudonáhodných čísel, baterie testů, NIST, šum, bezpečnost.

ABSTRACT

The bachelor thesis deals with the design of a hardware generator of random numbers using a low-power microcontroller MSP430. At first are described the existing methods of generating random numbers and comparison of different types generators. After that are stated the batteries of statistical tests which can be verified the generator on the basis of generated sequence. In the end of the thesis is to verify own proposals of the generator by using the statistical tests NIST.

KEYWORDS

True random number generator, pseudo-random number generator, battery of tests, NIST, noise, safety.

ECLER, Tomáš *Náhodný generátor čísel na bázi šumu pro nízkovýkonové zařízení* : bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2015. 43 s. Vedoucí práce byl Ing. Petr Mlýnek, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Náhodný generátor čísel na bázi šumu pro nízkovýkonové zařízení “ jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil(a) autorská práva třetích osob, zejména jsem nezasáhl(a) nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom(a) následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Petru Mlýnkovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Dále také Ing. Radku Fujdiakovi za pomoc při vypracování práce.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této bakalářské práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .
podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V současné době je stále aktuálnější otázka informační bezpečnosti různých informačních systémů, především pak bezpečnosti dat a komunikace [1]. Pokud by neexistovaly žádné metody zabezpečení, bylo by velice jednoduché zneužívat nechráněná data k nelegálním účelům. Obecně se pro tvorbu mechanizmů a algoritmů za účelem zabezpečení a utajení obsahu zpráv zabývá kryptografie.

V různých kryptografických aplikacích, jako například při elektronickém podpisu jsou využívána náhodná čísla, protože kryptografie potřebuje náhodná čísla, například ke generování parametrů pro distribuci klíčů, nebo ověření identity. Abychom získali náhodná čísla, je zapotřebí generátoru náhodných čísel, který se také využívá pro simulace, numerickou analýzu či loterii.

Náhodnost a vlastnosti generátoru mohou záviset, jak na náhodnosti fyzikálních procesů využívaných u hardwarových generátorů, tak i u softwarových generátorů pracujících na principu určitého algoritmu. Hlavním rozdílem je způsob provedení získání náhodných čísel. Softwarové generátory jsou označovány jako pseudonáhodné, jelikož náhodnost čísel se počítá podle některých algebraických funkcí nebo z určitých tabulek, kde se jejich model náhodnosti může opakovat. Hardwarové generátory používají jako zdroj vytvoření náhodných čísel některý fyzikální jev. Tento fyzikální jev by měl být nepředvídatelný, jako například atmosférický šum či rozpad radioaktivních částic, avšak výsledná čísla nemusí být vždy náhodná, proto se provádí kontrola náhodnosti čísel za pomoci různých baterií statistických testů.

Aby byl generátor využitelný v oblastech, ve kterých nelze využít externí zdroj energie, je možná implementace za pomoci nízko-výkonových zařízení, které jsou lehce přenositelné s nízkým odběrem energie.

Tato práce se zabývá popisem metod generování náhodných čísel a vlastní implementací návrhu hardwarového generátoru náhodných čísel, který je založený na neperiodických vlastnostech šumu s využitím nízko-výkonového mikrokontroleru. Následně je ověřena bezpečnost implementovaného generátoru za pomoci baterie testů NIST.

1 BEZPEČNOST

Bezpečnost – vlastnost nějakého systému, která vyjadřuje míru odolnosti vůči možným hrozbám a škodám. Při návrhu a zavádění bezpečnosti informací, je vhodné vzít v úvahu následující tvrzení [2]:

- Bezpečnost záleží především na lidech.
- Technologie jsou jen nástroje k prosazení bezpečnosti.
- Celková bezpečnost je určena nejslabším článkem v řetězci bezpečnostních opatření.
- Bezpečnost je dynamický proces, který podléhá neustálému vývoji a hodnocení.
- Nelze slučovat funkce výkonné a kontrolní ve všech fázích životního cyklu systému.
- Uživatelům je vhodné přidělovat jen minimální pravomoci nezbytné pro jejich práci.
- Bezpečnost systému musí být zachována i po obnově provozu systému.

1.1 Informační bezpečnost

Informační bezpečnost je souhrnné označení pro komplexní přístup k ochraně informací, informačních a komunikačních technologií. Cílem informační bezpečnosti je zejména ochrana informací a dat před negativními událostmi, jako je jejich ztráta, odcizení, zneužití, zničení, narušení, změn, tedy jakékoliv porušení celistvosti, důvěrnosti nebo dostupnosti. Rizika úniku či zneužití informací hrozí nejen z okolního prostředí, ale zejména zevnitř organizace samotné. Informační bezpečnost je součástí celoorganizačního řízení bezpečnosti a bezprostředně navazuje na ICT (počítačovou) bezpečnost, která se zabývá bezpečností informačních a komunikačních technologií. Z tohoto důvodu se k ochraně informací využívá kryptografie, což je věda a technika s úmyslem skrývání informací. Zajišťuje především důvěrnost a autentičnost přenášených zpráv. Za pomocí šifry můžeme vytvořit utajenou zprávu, kde veřejným klíčem se zpráva zašifruje a privátním dešifruje. Čím delší klíč, tím delší doba prolomení tohoto klíče potenciálním útočníkem. K vygenerování šifrovacího klíče můžeme využít stávajících generátorů náhodných čísel, ke kterým může být využit mikrokontroler s omezeným výkonem a aditivními moduly, pokud jej chceme využít na místech, kde není zaručen pevný přívod elektrické energie [3].

2 GENEROVÁNÍ NÁHODNÝCH ČÍSEL

Generování náhodných čísel není pouze klíčovou otázkou v kryptografických aplikacích, ale slouží i jako způsob protiopatření proti útokům, například na čipové karty, bezpečnostní protokoly, mikroprocesory, senzory, bezdrátové sítě a mnoho dalších zařízení může záviset na generování náhodných čísel. Generátor náhodných čísel musí splňovat přísné požadavky, vzhledem k tomu, že bezpečnostní protokoly závisí na nepředvídatelnosti klíče. Případný útočník nesmí být schopen předvídat posloupnost generovaných čísel, i za předpokladu znalosti konstrukce generátoru. Posloupnost generovaných čísel by měla být statisticky nezávislá.

2.1 Náhodné číslo

Náhodná čísla jsou čísla, které se splňují následující podmínky [4]:

1. Hodnoty jsou rovnoměrně rozloženy v definovaném intervalu.
2. Nemožnost předpovědět budoucí hodnoty na základě minulých či současných hodnot.

Jsou to sekvence celých čísel nebo skupiny čísel, které nevykazují naprosto žádný vztah k sobě navzájem v v celé posloupnosti čísel. Proto mnoho statistických metod spoléhá na takové náhodná čísla. Náhodná čísla jsou důležitá v mnoha oblastech, především v:

- modelování a simulacích,
- numerické analýze,
- rozhodování,
- počítačové grafice,
- kryptografii,
- datové komunikaci.

2.2 Typy generátorů náhodných čísel

Správný generátor náhodných čísel by měl splňovat následující vlastnosti:

1. Kvalitu – dodržení statistických vlastností testů, rovnoměrné rozložení 1 a 0.
2. Efektivnost – rychlost generování, spuštění a ukládání čísel, velkou délku sekvence čísel.
3. Opakovatelnost – výchozí bod pro opakovatelnost experimentu (generování čísel).
4. Přenositelnost – nezávislost systému k vytvoření kvalitních výsledků.

5. Jednoduchost – generátor by měl být jednoduchý pro implementaci s rychlým časem generování.

Abychom byli schopni vygenerovat náhodnou sekvenci čísel je zapotřebí využít generátoru náhodných čísel. Rozlišujeme dva typy generátorů [5]:

- PRNG – Pseudo-Random Number Generator česky „Pseudonáhodný generátor náhodných čísel“.
- TRNG – True Random Number Generator česky „Generátor skutečně náhodných čísel“.

2.2.1 Pseudonáhodný generátor náhodných čísel

Generátor pseudonáhodných čísel takzvaný „PRNG“, používá jeden nebo více vstupů a vytvoří tak více „pseudonáhodných“ čísel. Vstupy do PRNG se nazývají seeds česky „semínka“, které sama o sobě musí být náhodná a nepředvídatelná, což se nedá zaručit, proto je vhodnější pro výchozí vstupy „semínka“ generátoru PRNG využít výstupy generátoru TRNG, čímž se zaručí jejich náhodnost a nepředvídatelnost. Je-li pseudonáhodná sekvence správně vytvořena, každá další hodnota v pořadí se vytvoří z předchozí hodnoty pomocí transformací, které se zdají, že zavádí další náhodnost. Řada těchto transformací může eliminovat statistické autokorelace mezi vstupem a výstupem, z čehož lze vyvodit, že výstupy PRNG mohou mít lepší statistické vlastnosti a generovat sekvence čísel rychleji než TRNG [6] [7].

Aritmetické generátory

Využívají se pro účely počítačové simulace. Náhodná čísla jsou tvořena aritmetickými procedurami pomocí výpočtů, v nichž následující číslo závisí na jednom či více předchozích číslech. Nejčastěji se využívají lineární zpětnovazební registry. Jelikož jde o výpočet, nikoliv o náhodu, lze čísla takto získaná označit pouze za čísla pseudonáhodná. Pokud se vhodně zvolí aritmetická operace, tak se posloupnost vypočtených čísel po ověření náhodnosti jeví jako posloupnost opravdu náhodná. Pravda je však taková, že pseudonáhodná čísla můžou po nějaké době začít vykazovat periodicitu, tento problém je však minimální, protože inicializace posloupnosti by měla probíhat častěji, než by se periodicita stihla projevit. Periodicitu lze také eliminovat, pokud využijeme více vstupních „semínek“ ke generování, nejlépe výstupních hodnot generátoru TRNG. Aritmetické generátory mají v dnešní době využití převážně v kryptografii. Nejlepším kryptografickým algoritmem, jehož bezpečnost je matematicky dokázána je Vernamova jednorázová tabulka.

2.2.2 Skutečný generátor náhodných čísel

Generátor náhodných čísel, takzvaný „TRNG“, umožňuje generovat skutečně náhodná čísla, avšak může být citlivý na změnu prostředí. Sekvence čísel produkované těmito generátory mohou být nedostatečné pro vyhodnocení statickými testy a také vygenerování vysoce kvalitních sekvencí čísel může být časově náročné, jelikož jsou tyto generátory pomalejší než pseudonáhodné generátory. Výstup takového generátoru může být také využit, pro další zpracování generátorem PRNG, který může být vhodnější pro rychlejší vygenerování sekvence.

Mechanické generátory

Nejstarší typy generátorů, které jsou nejvíce známé mezi lidmi, jsou to například:

- hrací kostky,
- házení mincí,
- losování,
- tahání párátek.

Nejsou vhodné pro počítačové simulace.

Fyzikální a chemické generátory

Jsou to generátory ke generování náhodných a nedeterministických posloupností čísel. Posloupnost čísel je čistě náhodná, jelikož využívají chemických či fyzikálních pochodů, které mají náhodný charakter, pokud je neovlivňuje uměle vytvořený šum okolí. Nejčastější vstupní semínko pro hardwarové generátory se používá:

- **Atmosférický šum** – zdrojem atmosférického šumu jsou například bouřky. Atmosférický šum je poměrně snadné získat z běžných radiových vln, avšak je nutné se vyvarovat zdrojům, které by do měření vnášely prvky periodičnosti.
- **Rozpadu radioaktivních částic** – využití rozpadu radioaktivního zdroje, je dobrý způsob jak získat náhodná čísla. Čas rozkladu je zcela nepředvídatelný. K rozpadu dochází v náhodných okamžicích a tato nepředvídatelnost se využívá ke generování čísel. Měření probíhá mezi dvěma časovými impulzy, pokud jsou hodnoty stejné, proběhne nové měření.
- **Fotografie lávových lamp** – k vygenerování náhodných čísel využívá snímky lávové lampy. Avšak tato metoda se již nepoužívá.

  Náhodné sekvence čísel lze také získat bez použití hardwarových zařízení, náhodné hodnoty můžeme získat pomocí:

  – systémových hodin,

- ze stisku kláves či prodlevy mezi jednotlivými stisky,
- z obsahu vstup/výstupních bufferů,
- ze zátěže operačního systému,
- z analýzy síťového provozu,
- ze zvuku snímaného mikrofonem,
- z obrazu webkamery,
- z pohybu myší,
- z aktuální barvy některého z bodů na monitoru.

Abychom byli schopni realizovat kvalitní generátor náhodných čísel, bez jakékoliv opakující se periody, je nejlépe využít fyzikální či chemické generátory, ve kterých se jako nejčastější entropie využívá šum [8] [9] [10].

2.2.3 ŠUM

Šum je definován jako jakýkoliv nechtěný či nežádoucí náhodný signál, který lze rozdělit:

Vnější zdroj šumu – vytvořený zcela nezávisle, lze se s ním setkat všude kolem nás, například atmosférický šum, který se vyskytuje v životním prostředí z různých přírodních jevů, jako jsou bouře, vítr, moře apod. Šumem jsme ovlivnění i z vesmíru, například paprsky slunce, či vlivem vesmírného prachu dopadajícího na Zemi.

Uměle vytvořený zdroj šumu – nechtěně vytvořený zdroj, například v elektronických obvodech, který následně můžeme dělit do následujících kategorií:

v elektronice – takzvaný tepelný šum existuje ve všech obvodech a zařízeních, jehož důsledkem je tepelná energie. Čím vyšší teplota, tím vyšší šum,

v oblasti telekomunikačních služeb – šum vznikne při přenosu signálu, většinou závisí na způsobu přenosu, použitých přenosových médií a prostředí, ve kterém se přenáší, v oblasti průmyslu šum z letadel, automobilů, celkově z elektrických motorů, spínacích zařízení, vysokého napětí, zářivek apod [11].

Bílý šum

Šum se podle rozložení výkonu ve spektru dělí na různé druhy, bílý šum je náhodný signál s rovnoměrnou výkonovou spektrální hustotou, který lze nejlépe využít ke generování náhodných čísel, jelikož nemá žádnou autokorelaci, což znamená nulový odhad dalších generovaných čísel. Důvod pojmenování tohoto druhu šumu je takový, že jeho spektrální hustota je stejná na všech frekvencích, což vyjadřuje skutečnost, že jednotlivé spektrální složky šumu jsou frekvenčně nezávislé podobně jako složky bílého světla ve viditelném spektru.

2.2.4 Druhy šumu využitelné ke generování čísel

Šum lze dělit podle využitelnosti z hlediska generování čísel, nejčastěji se ke generování používají vyhřívané rezistory, závěrně polarizované přechody báze či závěrně polarizované přechody Zenerovy diody.

Tepelný šum

Vodiče obsahují velký počet volných elektronů a iontů, které svou vibrací vytváří teplo, nejčastějším případem je zdroj odporu ve vodiči, kde pohyb volných elektronů generuje proud, který má čistě náhodný charakter. Spektrální výkonová hustota nezávisí na frekvenci, proto se teplotní šum řadí do kategorie bílého šumu.

Výstřelový šum

Vyskytuje se u součástek s PN přechodem vlivem kvantové generace a rekombinace náboje, za předpokladu že jím protéká proud. Výsledný šum vzniká pohybem nabitých částic.

Blikavý šum

Je způsoben poruchami v krystalové mřížce a nečistot v polovodiči. Spektrální hustota výkonu klesá s rostoucím kmitočtem, označuje se jako šum. Projevuje se především na nižších kmitočtech.

Praskavý šum

Vzniká vlivem znečištění přechodu mezi bází a emitorem ionty těžkých kovů. Spektrální hustota výkonu klesá s rostoucím kmitočtem, jedná se opět o šum [12].

2.2.5 Porovnání TRNG a PRNG

Z porovnání obou typů generátorů dojdeme k závěru, že z omezenosti jednoho typu generátoru se stávají výhodou druhého generátoru a naopak, což dokazují následné výhody a nevýhody těchto TRNG, kde pro PRNG je to přesně naopak. Porovnání základních vlastností kvality generátorů lze pozorovat v tabulce 2.1 [4].

Výhody:

- Vysoký stupeň zabezpečení.
- Nemožnost předpovědět následující čísla na základě znalosti předcházejících čísel.
- Neexistuje žádná závislost v rámci postupnosti posloupnosti.

- Nemá žádné cyklické postupnosti.

Nevýhody:

- Nízká rychlost generování náhodných čísel.
- Dražší pořizovací cena.
- Potřeba zabezpečení proti útokům na fyzické úrovni.
- Opakované vygenerování stejné posloupnosti.

Tab. 2.1: Zhodnocení generátorů [4]

VLASTNOST	**PRNG**	**TRNG**
VÝKONNOST	Vysoká	Nízká
DETERMINISTIČNOST	Deterministický	Nedeterministický
PERIODICITA	Periodický	Neperiodický

2.3 Metoda generování náhodných čísel pomocí LSB

Abychom získali posloupnost generovaných čísel ve tvaru jedniček a nul, je zapotřebí využít některou z metod transformace. V této práci bude využita metoda nejméně významného bitu (LSB), která určuje, zdali je hodnota posledního bitu čísla v binární podobě sudá nebo lichá.

Nejméně významný bit je nejnižší bit v sérii binárních čísel, nachází se na konci řetězce ⇒ 10010110 **1**. Série binárních čísel nám udává hodnotu jednotky, kde se určí pomocí nejméně významného bitu, zdali je poslední bit sudý nebo lichý. Pokud je bit sudý, bude binární hodnota 0, pokud lichý tak naopak. Máme například decimální hodnotu 3366, v binárním tvaru má hodnotu 110100100110, hodnota posledního bitu je sudá, nejméně významný bit bude binární hodnota 0. Pomocí početní operace modulo, která souvisí s celočíselným dělením si můžeme ověřit hodnotu posledního bitu. Operace je určena vzorcem

$$(a \ \ mod \ \ m) = a - \left\lfloor \frac{a}{m} \right\rfloor m, \tag{2.1}$$

kde a je celé číslo, které chceme dělit a m je dělitel. Výslednou hodnotou po celočíselném dělení je zbytek děleného čísla [13].

2.4 Statistické testování náhodnosti generovaných čísel

Důležitou součástí generování náhodných čísel je kvalifikovat generátory náhodných čísel pomocí statických testů. Statické testy nám určují, zdali testovaná sekvence bitů vyhoví či nevyhoví daným podmínkám testu. Porovnávají se tři hlavní body ke splnění podmínek statických testů:

- Statistický údaj zkoušek pro binární sekvence nesmí klesnout pod danou prahovou úroveň významnosti α.
- Počet 1 a 0 musí odpovídat polovině rozsahu dané sekvence.
- Výsledná hodnota pravděpodobnosti (P-hodnota) bitové sekvence testu musí být rovna nebo vyšší než volená kritická hodnota α.

P-hodnoty jsou vypočítány pomocí speciálních funkcí:

1. Kumulativní rozdělení pravděpodobnosti

$$\phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-u^2/2} \, \mathrm{d}u. \tag{2.2}$$

2. Doplňkové chybová funkce **ercf**

$$erfc(z) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{z} e^{-u^2} \, \mathrm{d}u. \tag{2.3}$$

3. Neúplné gama funkce **igamc**

$$Q(a,x) \equiv 1 - P(a,x) \equiv \frac{\Gamma(a,x)}{\Gamma(a)} \equiv \frac{1}{\Gamma(a)} \int_{x}^{\infty} e^{-t} t^{a-1} \, \mathrm{d}t, \tag{2.4}$$

   kde $Q(a,0) = 1$ a $Q(a,\infty) = 0$.

Aby byl test ohodnocen jako úspěšný, musí být splněna testovací hypotéza H_0, proti níž stojí alternativní hypotéza H_A, která popírá hypotézu H_0. Testovací hypotéza H_0 je splněna, pokud vypočtená P-hodnota splňuje podmínku $P-hodnota \geq \alpha$, kde kritická hodnota α se většinou volí 0,1, 0,05, 0,01 či 0,001. V mé práci byla zvolena kritická hodnota $\alpha = 0{,}1$. Pokud je $P - hodnota \leq \alpha$, tak platí alternativní hypotéza H_A, která zamítá testovací hypotézu H_0. Čím nižší je kritická hodnota, tím obtížnější je zamítnutí H_0, avšak na úkor ztráty kvality generované posloupnosti.

Platí-li však hypotéza H_0, ale na základě výběru kritické hodnoty ji zamítáme, tak se dopustíme chyby prvního druhu. Jestliže hypotéza H_0 neplatí, ale na základě výběru kritické hodnoty ji nezamítáme, dopustíme se chyby druhého druhu [14].

Myšlenka statistického testování náhodně generované posloupnosti, vznikla jako reakce na množství kryptografických útoků, které využívali opakovaných závislostí

v datech, což by skutečně vygenerovaná data neměli mít. Vzniklo proto mnoho baterií testů, ale i mnoho samostatných testů, kde každá posloupnost bez ohledu na způsobu vytvoření, může být testována libovolným testem [15].

2.4.1 Baterie statistických testů

Generátory náhodných sekvencí čísel, jak už pseudonáhodné PRNG tak i skutečné TRNG je nutné otestovat, zdali jejich výstupní sekvence bitů 1 a 0 jsou opravdu náhodné. Existuje mnoho testovacích programů, nejznámější z nich jsou DieHardy a ENT, avšak jsou již zastaralé, jelikož jejich vývoj již nepokračuje. Proto NIST publikovala statistickou testovací sadu celkem 15 testů, jejichž software se každým rokem vyvíjí do stále lepší podoby, s cílem vyzkoušet náhodnost výstupní binární sekvence skutečných i pseudonáhodných generátorů s minimální chybovostí. Všechny jsou vyhodnoceny jako funkční, popisující podmínky výběru a testování náhodných a pseudonáhodných čísel generátorů. Tyto statické testy porovnávají a vyhodnocují posloupnosti náhodného pořadí, ale mohou také sloužit k určení, zda je generátor vhodný pro konkrétní kryptografickou aplikaci. Během testu se výsledná hodnota porovnává s kritickou hodnotou, která je pro daný test definována. Ve většině případů by výsledná hodnota zkoumané testované sekvence bitů měla být větší či rovna kritické hodnotě 0,01, pokud tak nastane, je sekvence bitů uznána za náhodnou. Je-li kritická hodnota překročena, tak to znamená, že sekvence bitů není náhodná.

DieHardy

První komplexní softwarové řešení statistických testů, které vzniklo jako reakce na naprostý nedostatek testovacích nástrojů. Hlavní problém u těchto testů je, že samotné testy jsou přednastavené napevno, tudíž nelze měnit jejich parametry. Vyžaduje pevně daný formát vstupního souboru s předdefinovanými daty, což prakticky znamená, že testuje malé vzorky vstupních dat, a to vede k nekvalitnímu otestování. Avšak velkou výhodou této baterie testů je rozšiřitelnost o další statistické testy [16].

ENT

Je jednoduchý program s již naimplementovanou baterií testů. Určen k testování sekvencí bitů uložených v souborech. Program je užitečný pro vyhodnocení generátorů náhodných čísel z hlediska šifrování, statistických aplikací a kompresních algoritmů. Ent provádí řadu testů proudu bajtů, jejichž vyhodnocením je:

- Entropie – vyjádření počtu bitů na znak (obrazový soubor).
- Optimální komprese pro zmenšení velikosti.

- Chi kvadrát – rozdělení do vzorků, které nesmí překročit nastavenou kritickou hodnotu.
- Aritmetický průměr – součet všech bitů v testovaném souboru , následné vydělení všech hodnot v daném souboru, výsledkem je průměrná odchylka od požadované kritické hodnoty [17].

NIST

Nejmodernější testovací program náhodnosti, dříve to byl DieHardy, avšak již není udržován. Nist rámec, stejně jako mnoho dalších testů je založen na testování hypotéz, který obsahuje výběr z nejlepších dosavadních testů a několik nových řešení. Je to první volně dostupný program s grafickým rozhraním, umožňující pohodlné testování. Program obsahuje sadu předprogramovaných pseudo-generátorů, a umožňuje i testování v případě vstupu dat ze souboru, proto se stal standardem na poli testování. Jelikož testy od NIST využívám ve své práci k otestování, zda je konkrétní testovaná posloupnost 0 a 1 náhodná, tak budou tyto testy blíže popsány [18].

2.5 Baterie testů NIST

Existuje potenciálně nekonečně velké množství různých statistických testů,kde každý test testuje nějakou specifickou vlastnost, kterou generátor náhodných čísel nemusí splnit, čímž je označený za nenáhodný. Právě proto vznikla snaha o vytvoření co nejmenší množiny testů, která by byla vhodná na efektivní a téměř bezchybné rozhodování o náhodnosti testovaných dat.

2.5.1 Frekvenční test

Prostřednictvím tohoto testu je určeno, zdali je frekvence 1 a 0 v testované posloupnosti přibližně stejná, podíl 1 a 0 by se měl blížit jedné polovině, což se očekává pro skutečně náhodnou sekvenci. Každý bit 0 a 1 je reprezentován jako -1 a 1 pomocí matematického vztahu $X_i = 2\epsilon_i$-1, kde X_i představuje novou hodnotu a ϵ_i bitový blok určené hodnoty.

Popis zkoušky:

1. Konverze vstupní posloupnosti na hodnoty $0 \Rightarrow -1; 1 \Rightarrow 1$, které jsou následně sečteny do výsledku Sn.
2. Výpočet statistické hodnoty $s_{obs} = \frac{Sn}{\sqrt{n}}$, kde n je délka bloku bitů.
3. Výpočet $P - hodnoty = \mathbf{erfc}\left(\frac{s_{obs}}{\sqrt{2}}\right)$, kde kde **erfc** je doplňková chybová funkce.

2.5.2 Frekvenční test v rámci bloku

Tímto testem se porovnává podíl rovnoměrného rozložení 1 a 0 v celé M-bitové posloupnosti. Posloupnost je rozdělena do několika nepřekrývajících se M-bloků po 10, v nichž by mělo být rovnoměrné rozložení 1 a 0. Přebytečné bity se zahazují.

Popis zkoušky:

1. Rozdělení vstupní posloupnosti bitů do $N = \left\lfloor \frac{n}{M} \right\rfloor$, kde n je délka řetězce bitů a M délka každého bloku.
2. Stanovení poměru jedniček v každém bloku $\pi = \frac{\sum_{j=1}^{M} \epsilon_{(i-1)M+j}}{M}$, pro $1 \leq$ i $\leq$ N.
3. Výpočet statistiky $\chi^2(obs) = 4\mathrm{M}\sum_{i=j}^{N}(\pi_i - \frac{1}{2})^2$.
4. Výpočet $P - hodnoty =$ **igamc**$(\frac{N}{2})$, $(\frac{\chi^2(obs)/2}{2})$, kde **igamc** je nekompletní funkce $Q(a, x) = 1 - P(a, x)$.

2.5.3 Test série bitů

Cílem testu je nepřetržitý sled bitů o stejné velikosti, ve kterém se zaměřuje na celkový počet 0 a 1 v celé řadě. Účelem testu je zjistit, zda je počet 1 a 0 různých délek po sobě jdoucích různorodý, výsledkem je zjištění, zda je oscilace mezi 1 a 0 dostatečně velká, aby odpovídala náhodné sekvenci.

Popis zkoušky:

1. Výpočet podílu jedniček ve vstupní sekvenci $\pi = \frac{\sum j \epsilon j}{n}$.
2. Kontrola požadované frekvence $|\pi - \frac{1}{2}| \geq \tau$
3. Výpočet statistiky $Vn(obs) = \sum_{k=1}^{n-1} r(k) + 1$, kde $r(k) = 0$, za předpokladu $\epsilon k = \epsilon k + 1$, jinak $r(k) = 1$.
4. Výpočet $P - hodnoty =$ **erfc** $\left(\frac{|V_n(obs) - 2n\pi(1-\pi)|}{2\sqrt{2n}\pi(1-\pi)}\right)$.

2.5.4 Test na nejdelší běh jedniček v bloku

Zaměření testu je na nejdelší běh 1 v rámci M–bitového bloku. Účelem zkoušky je zjistit, zdali délka nejdelšího běhu 1 v testované sekvenci, kde minimální testovaná sekvence je dána tabulkou 2.2, je v souladu s očekávaným počtem pro náhodné sekvence.

n = délka posloupnosti
M = délka sekvence bloku

Popis zkoušky:

Tab. 2.2: Minimální sekvence.

n	**M**
128	8
6272	128
750000	10^4

1. Rozdělení pořadí na M-bloky.
2. Rozdělení do tabulky podle kategorie nejdelší série jedniček v každém bloku.
3. Výpočet statistiky $\chi^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, kde hodnoty K a N jsou dány hodnotou M podle tabulky 2.3.
4. Výpočet $P - hodnoty = \mathbf{igamc}\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right)$

Tab. 2.3: Hodnoty sekvence

M	**K**	**N**
8	3	16
128	5	49
10^4	6	75

2.5.5 Test sérií binárních matic

Test má za úkol zjistit, zdali se v celé bitové posloupnosti neopakují vzory po obě jdoucích bitů. Bitová posloupnost je rozdělena na nesouvislé bloky, kde každý blok je reprezentován obvykle maticí 32 řádků a 32 sloupců, zbývající nepoužité bity jsou zahozeny.

Popis zkoušky:

1. Posloupnost bitů rozdělena do nesouvislých maticových bloků M a Q, kde $N = \left\lfloor \frac{n}{MQ} \right\rfloor$
2. Určení binárního pořadí (R_l) každé matice, kde $l = 1, \ldots, N$.
3. Určení matic
   $F_M \Rightarrow R_1 = M$ (plná hodnost matic),
   $F_M - 1 \Rightarrow R_1 = M - 1$ (počet -1 v matici),
   $N - F_M - F_M - 1$ (počet zbývajících matic).

4. Výpočet statistiky $\chi^2(obs) = \frac{(F_M - 0{,}2888N)^2}{0{,}2888N} + \frac{(F_{M-1} - 0{,}5776N)^2}{0{,}5776N} + \frac{(N - F_M - F_{M-1} - 0{,}1336N)^2}{0{,}1336N}$.
5. Výpočet $P - hodnoty = e^{\frac{-\chi^2(obs)}{2}}$.

2.5.6 Test Diskrétní Fourierovy transformace

Pomocí testu se určuje, zda má bitová sekvence pravidelné rysy napříč celou posloupností. Pravidelné rysy můžeme chápat jako opakující se vzory, které jsou blízko u sebe. Test provádí diskrétní Fourierovy transformace, kde každý bit v pořadí má svoji výšku, podle nastavené prahové hodnoty se zjistí počet všech stejných vrcholů, který nesmí přesáhnout 5 %, čímž lze sekvenci označit za náhodnou.

Popis zkoušky:

1. Konverze vstupní posloupnosti na hodnoty $0 \Rightarrow -1; 1 \Rightarrow 1$.
2. Použití Diskrétní Fourierovy (DFT) $S = DFT(X)$.
3. Výpočet $M = modulus(S') \equiv |S'|$, kde S' je podřetězec skládající se z prvních $n/2$ prvků v S a funkce modulus produkuje posloupnost hodnot výšek.
4. Výpočet $T = \sqrt{3n}$, kde T je maximální výška prahové hodnoty, která by neměla překročit hodnotu 95 %.
5. Výpočet $N_0 = \frac{0{,}95n}{2}$, což je očekávaná teoretická hodnota 95 % za předpokladu náhodnosti.
6. Výpočet N_1 hodnoty, což je skutečná hodnota pozorovaných výšek v celé posloupnosti.
7. Výpočet $d = \frac{(N_1 - N_0}{\sqrt{\frac{n(0{,}95)(0{,}05)}{2}}}$.
8. Výpočet $P - hodnoty = \mathbf{erfc}\left(\frac{|d|}{\sqrt{2}}\right)$.

2.5.7 Test nepřekrývajících se vzorů

Tento test zjišťuje v posloupnostech příliš mnoho výskytů aperiodických vzorů. Test hledá výskyty předem zadané bitové sekvence, a zdali se nepřekrývají, počet těchto událostí musí být v mezích statistické posloupnosti za předpokladu náhodnosti. U testu se používá takzvané m–okno, které vyhledává konkrétní m–vzor, když tento m–vzor není nalezen, m–okno se posouvá o pozici jednoho bitu dále, pokud je nalezen, m–okno se posouvá o následující bit po nalezeném vzoru a hledání pokračuje.

Popis zkoušky:

1. Rozdělení posloupnosti do N nezávislých bloků délky M.
2. Počet případů shody v rámci $m - okna$.
3. Výpočet střední hodnoty $\mu = \frac{(M-m+1)}{2^m}$ a rozptylu $\sigma^2 = M(\frac{1}{2^m} - \frac{2m-1}{2^{2m}})$.
4. Výpočet statistiky $\chi^2(obs) = \sum_{j=1}^{N} \frac{(W_j-\mu)^2}{\sigma^2}$.
5. Výpočet $P - hodnoty = \mathbf{igamc}\left(\frac{N}{2}, \frac{\chi^2(obs)}{2}\right)$.

2.5.8 Test překrývajících se vzorů

Test je v zásadě podobný jako předchozí test nepřekrývajících se vzorů s tím rozdílem, že pokud je m–vzor nalezen, tak se m-okno posune o jeden bit před dalším pokračováním hledání. Přebytečné bity se zahazují.

Popis zkoušky:

1. Rozdělení posloupnosti do N nezávislých bloků délky M.
2. Počet případů shody v rámci $m - okna$.
3. Výpočet hodnot $\lambda = \frac{(M-m+1)}{2^m}$; $\eta = \frac{\lambda}{2}$.
4. Výpočet statistiky $\chi^2(obs) = \sum_{i=0}^{5} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, kde $\pi_0 = 0,367879$, $\pi_1 = 0,183940$, $\pi_2 = 0,137955$, $\pi_3 = 0,099634$, $\pi_4 = 0,069935$, $\pi_5 = 0,140657$.
5. Výpočet $P - hodnoty = \mathbf{igamc}\left(\frac{5}{2}, \frac{\chi^2(obs)}{2}\right)$.

2.5.9 Mauerův univerzální statický test

Cílem testu je zjistit, zdali mohou být mezi sebou bity komprimovány bez ztráty informací, myslí se tím minimalizovat stejné bity v jednom M-bloku. Značně stlačitelné sekvence jsou označeny za nenáhodné. Testování se většinou provádí v bloku M o deseti bitech. Přebytečné bity se zahazují.

Popis zkoušky:

1. Bitová posloupnost je rozdělena do dvou nepřekrývajících se bloků označených Q a K. Q bloky slouží k inicializaci a bloky $K(K = \left\lfloor \frac{n}{L} \right\rfloor - Q)$ jako testovací, kde L je délka bloku většinou o dvou bitech.
2. Počet výskytů stejných L bloků.
3. Přidaní vypočtené vzdálenosti mezi výskyty stejných L bloků součtem všech zjištěných rozdílů v K blocích $\Rightarrow sum = sum + log_2(i - T_j)$.

4. Výpočet statistiky $f_n = \frac{1}{K}\sum_{i=Q+1}^{Q+K} log_2(i - T_j)$, kde T_j je tabulka odpovídající desítkovému vyjádření obsahu L bloku.
5. Výpočet $P - hodnoty = \mathbf{erfc}\left(\left|\frac{f_n - očekávaná\ \ hodnota(L)}{\sqrt{2}\sigma}\right|\right)$.

2.5.10 Test lineární složitosti

Cílem testu je zjistit četnost všech možných překrývajících se M-bitových vzorků v bloku deseti bitů v celé posloupnosti. Princip toho testu je podobný jako u testu překrývajících se vzorků. Je možno přidat navíc 1 a 0, aby četnost dosahovala lepších výsledků.

Popis zkoušky:

1. Rozdělení posloupnosti n do N nezávislých bloků délky M, kde $n = MN$.
2. Určení lineární složitosti L_i každého N bloku, kde L_i je délka nejkratší posloupnosti posunu zpětnovazebního lineárního registru.
3. Výpočet střední hodnoty $\mu = \frac{M}{2} + \frac{(9+(-1)^{M+1})}{36} - \frac{\frac{M}{3}+\frac{2}{9}}{2^M}$.
4. Výpočet hodnoty $T_i = (-1)^M \bullet (L_i - \mu) + \frac{2}{9}$.
5. Výpočet statistiky $\chi^2(obs) = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$, kde $\pi_0 = 0,01047$, $\pi_1 = 0,03125$, $\pi_2 = 0,125$, $\pi_3 = 0,5$, $\pi_4 = 0,25$, $\pi_5 = 0,0625$, $\pi_6 = 0,02078$.
6. Výpočet $P - hodnoty = \mathbf{igamc}\left(\frac{K}{2}, \frac{\chi^2(obs)}{2}\right)$.

2.5.11 Test sérií

Cílem testu je zjistit četnost všech možných překrývajících se M-bitových vzorků v bloku deseti bitů napříč celou posloupností. Princip toho testu je podobný jako u testu překrývajících se vzorků. Je možno přidat navíc 1 a 0, aby četnost dosahovala lepších výsledků.

Popis zkoušky:

1. Rozšíření bloku bitů na konci sekvence o určitý počet m bitů připojených ze začátku sekvence.
2. Stanovení četnosti všech překrývajících se m–bitových bloků o velkosti tří bitů.
3. Výpočet
   $\psi^2_m = \frac{2^m}{n}\sum_{i_1}\ldots i_m(v_{i_1}\ldots i_m - \frac{n}{2^m})^2 = \frac{2^m}{n}\sum_{i_1\ldots ß_m} v_i^2 \ldots i_m - n,$
   $\psi^2_{m-1} = \frac{2^m-1}{n}\sum_{i_1}\ldots i_m(v_{i_1}\ldots i_m - \frac{n}{2^m-1})^2 = \frac{2^m-1}{n}\sum_{i_1\ldots ß_m} v_i^2 \ldots i_m - n,$
   $\psi^2_{m-2} = \frac{2^m-2}{n}\sum_{i_1}\ldots i_m(v_{i_1}\ldots i_m - \frac{n}{2^m-2})^2 = \frac{2^m-2}{n}\sum_{i_1\ldots ß_m} v_i^2 \ldots i_m - n.$

4. Výpočet
   $\nabla\psi_m^2 = \psi_m^2 - \psi_{m-1}^2$,
   $\nabla^2\psi_m^2 = \psi_m^2 - 2\psi_{m-1}^2 + \psi_{m-2}^2$.
5. Výpočet
   $P - hodnoty1 = \mathbf{igamc}(2^{m-2}, \nabla\psi_m^2)$,
   $P - hodnoty2 = \mathbf{igamc}(2^{m-3}, \nabla^2\psi_m^2)$.

2.5.12 Test entropie

Entropický test náhodnosti je založený na periodicitě vzorků. Cílem testu je zjistit četnost překrývajících se M-bitových vzorků v bloku deseti bitů napříč celou posloupností. Účelem je porovnat frekvenci dvou přilehle jdoucích překrývajících se bloků.

Popis zkoušky:

1. Rozšíření bloku bitů na konci sekvence o určitý počet m bitů připojených ze začátku sekvence.
2. Stanovení četnosti všech překrývajících se m–bitových bloků, většinou o třech bitech.
3. Výpočet $C_i^m = \frac{i}{n}$ pro všechny i, kde i je hodnota m–bloku .
4. Výpočet $\varphi^{(m)} = \sum_{i=0}^{2^n-1} \pi_i log\pi_i$, kde $\pi_i = C_j^3$ a $j = log_2 i$.
5. Opakování kroků 1 až 4 pro všechny hodnoty.
6. Výpočet statistiky $\chi^2(obs) = 2n[log2 - ApEn(m)]$, kde $ApEn(m) = \varphi^m - \varphi^{m+1}$.
7. Výpočet $P - hodnoty = \mathbf{igamc}(2^{m-1}, \frac{\chi^2}{2})$.

2.5.13 Test kumulačních součtů

Účelem testu je, zda kumulativní součet dílčích sekvencí je příliš velký či malý. Čím blíž je výsledek nule, tím lepší náhodnost.

Popis zkoušky:

1. Konverze vstupní posloupnosti na hodnoty $0 \Rightarrow -1; 1 \Rightarrow 1$.
2. Výpočet dílčí sumy po sobě jdoucích sekvencí $S_n \Rightarrow S_k = S_{k-1} + X_k$, pro mód 0, jinak$S_k = S_{k-1} + X_{n-k+1}$ pro mód 1.

3. Výpočet statistiky $z = max_{1 \leq k \leq n}|S_k|$, kde $max_{1 \leq k \leq n}|S_k|$ je největší z absolutních hodnot součtů S_k.

4. Výpočet
$P - hodnoty = 1 - \sum_{k=\frac{\frac{-n}{z}+1}{4}}^{\frac{\frac{n}{z}-1}{4}} \left[\phi\left(\frac{(4k+1)z}{\sqrt{n}}\right) - \phi\left(\frac{(4k-1)z}{\sqrt{n}}\right)\right] +$
$\sum_{k=\frac{\frac{-n}{z}-3}{4}}^{\frac{\frac{n}{z}-1}{4}} \left[\phi\left(\frac{(4k+3)z}{\sqrt{n}}\right) - \phi\left(\frac{(4k+1)z}{\sqrt{n}}\right)\right]$.

2.5.14 Test náhodných exkurzí

Zaměření testu je podobné jako u předchozího testu kumulačních součtů, vyhodnocuje se kumulativní součet. Test se skládá z více dílčích testů, pro které platí, že když jeden neprojde, nebude náhodný, tak bude celý test vyhodnocen jako neúspěšný, a tudíž ne-náhodný. Test vyhodnocuje kolik nul je v sekvenci bloku M o dvanácti bitech a na jaké pozici, počítá se i výpočet frekvence pro každou nulu v cyklu.

Popis zkoušky:

1. Konverze vstupní posloupnosti na hodnoty $0 \Rightarrow -1; 1 \Rightarrow 1$.

2. Výpočet dílčí sumy po sobě jdoucích sekvencí $S_n \Rightarrow S_k = S_{k-1} + X_k$, pro mód 0, jinak$S_k = S_{k-1} + X_{n-k+1}$ pro mód 1.

3. Vložení 0 před a za novou sekvenci $S' \Rightarrow S' = 0, s_1, s_2, \ldots, s_n, 0$.

4. Určení celkového počtu nulových přechodů v sekvenci S', kde J je počet cyklů v sekvenci s počáteční hodnotou 0 a koncovou hodnotou 0.

5. Určení počtu stejných výskytů hodnot v cyklu v intervalech $-4 \leq x \leq -1$ a $1 \leq x \leq 4$.

6. Pro každou hodnotu z intervalu se vypočte $v_k(x) =$ celkový počet cyklů v určité hodnotě intervalu.

7. Výpočet statistiky $\chi^2(obs) = \sum_{k=0}^{5} \frac{(v_k(x) - J\pi_k(x))^2}{J\pi_k(x)}$, kde $\pi_k(x)$ je pravděpodobnost, že pro různé hodnoty z intervalu dochází k časově náhodné distribuci.

8. Výpočet $P - hodnoty = \mathbf{igamc}\left(\frac{5}{2}, \frac{\chi^2(obs)}{2}\right)$.

2.5.15 Test náhodných exkurzních variant

Cílem testu je počet, kolikrát nastane stejný stav v kumulativním součtu. Princip testu je podobný jako u testu náhodných exkurzí. Cílem je zjistit odchylky od očekávaného počtu výskytů jednotlivých vzorů. Test se skládá z 18ti dílčích testů.

Popis zkoušky:

1. Konverze vstupní posloupnosti na hodnoty $0 \Rightarrow -1; 1 \Rightarrow 1$.
2. Výpočet dílčí sumy po sobě jdoucích sekvencí $S_n \Rightarrow S_k = S_{k-1} + X_k$, pro mód 0, jinak$S_k = S_{k-1} + X_{n-k+1}$ pro mód 1.
3. Vložení 0 před a za novou sekvenci $S' \Rightarrow S' = 0, s_1, s_2, \ldots, s_n, 0$.
4. Pro každou hodnotu z intervalu se vypočte $v_k(x) =$ celkový počet cyklů v určité hodnotě intervalu.
5. Výpočet $P - hodnoty = \mathbf{erfc}\left(\frac{|\varsigma(x)-J|}{\sqrt{2J(4|x|-2)}}\right)$ [18].

3 VLASTNÍ IMPLEMENTACE NÁHODNÉHO GENERÁTORU ČÍSEL

Pokud chceme použít generátor náhodných čísel, tak nejlepší využití najdeme právě ve skutečně náhodných generátorech čísel, který můžeme dále rozvíjet pseudonáhodným generátorem, například pro potřeby kryptografie. Pro uživatelsky přívětivější generování je možné využít nízkovýkonový mikrokontroler, díky své nízké spotřebě energie a FRAM energeticky nezávislé feroelektrické paměti na čipu. Výhodou je také přepnutí do režimu spánku, pokud zařízení v daný moment nepoužíváme.

3.1 Nízko-výkonové zařízení

Jsou to nízko energetické systémy malého rozměru, levné výroby s nízkou spotřebou MCU. Jsou oblíbené především díky své jednoduchosti, přenositelnosti a bezproblémové integraci s jinými zařízeními. Výhodou je sběr dat v reálném čase, proto se využívají v mnoha odvětvích, jako například:

- v energetice
  - měření na sloupech vedení vysokého a velmi vysokého napětí,
- ve zdravotnictví,
- v komunikačních technologiích,

u kterých je časová i paměťová náročnost velice podstatná, z hlediska výdrže baterie či rychlosti komunikace. Zařízení je možno použít také jako prostředek k připojení aditivního modulu, jako například teploměru nebo hardwarového generátoru. Díky svému nízko-energetickému režimu spánku či hibernace (pokud se v danou chvíli nevyužívá) i se zachováním obsahu RAM se prodlužuje životnost baterií, k čemuž je zařízení převážně určeno.

3.1.1 Nízko-výkonový mikrokontroler MSP430

Ve své práci používám nízko-výkonový mikrokontroler MSP430 řady MSP430x5xxx (konkrétně MSP430F5438A), který je vyvíjen firmou Texas Instrument. Tato řada mikrokontrolerů je schopna pracovat s frekvencí CPU až do 25 MHz, mají 512 kB flash paměti, a až 66 kB RAM. K uživatelsky přívětivějšímu ovládání mikrokontroleru (nahrávání softwaru, vymazání paměti) je možné využít nízko-výkonové zařízení MSP-FET430UIF, které nepotřebuje žádné externí napájení. Do počítače je toto zařízení připojeno pomocí USB. K implementaci programového kódu, který lze psát v jazyku C či assembler, slouží program Code Composer Studio, což je vývojový

software pro práci s produkty od firmy Texas Instruments, tedy i pro náš přípravek MSP430 [19] [20].

3.2 Program pro ovládání generátoru náhodných čísel

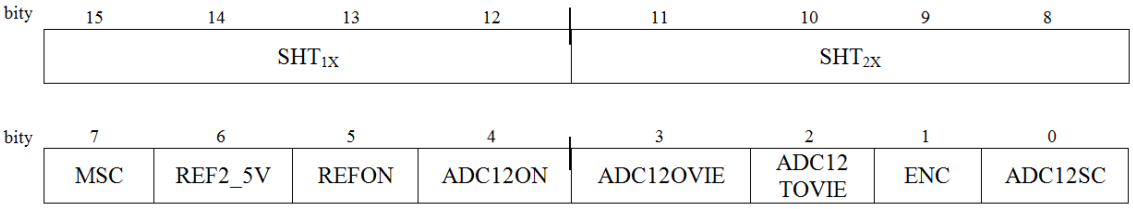
V této části bude popsán program, který generuje pravá náhodná čísla na bázi tepelného šumu s využitím analog-digitalního převodníku.

Proces WatchDog Timer slouží pro vypnutí či zastavení programu, působí jako ochrana mikrokontroleru proti zacyklení.

Ukázka kódu 3.1: WatchDog Timer

```
WDTCTL = WDTPW + WDTHOLD;
```

V následujících krocích již nastavujeme příslušné registry, především pak registr ADC12CTL0 pro funkci analog-digitálního převodníku mikrokontroleru k převedení analogové hodnoty výstupního pinu na digitální. Registr pro ADC12CTL0 můžeme vidět na obr. 3.1. Bity 0 a 1 slouží pro start a povolení konverze. Bity 2 a 3 jsou kontrolní bity, které znovu nastavují obsah ADC12SC pokud byl vymazán. Následující bity 4 až 15 se nastavují pouze pokud není povolena konverze, tedy musí platit podmínka ENC = 0. Bity 4 až 7 spínají a nastavují AD převodník s referenční hodnotu napětí. Bity 8 až 15 drží hodnotu 1 nebo 0 pomocí S/H, kterou následně ukládají do paměti ADC12MEM0.

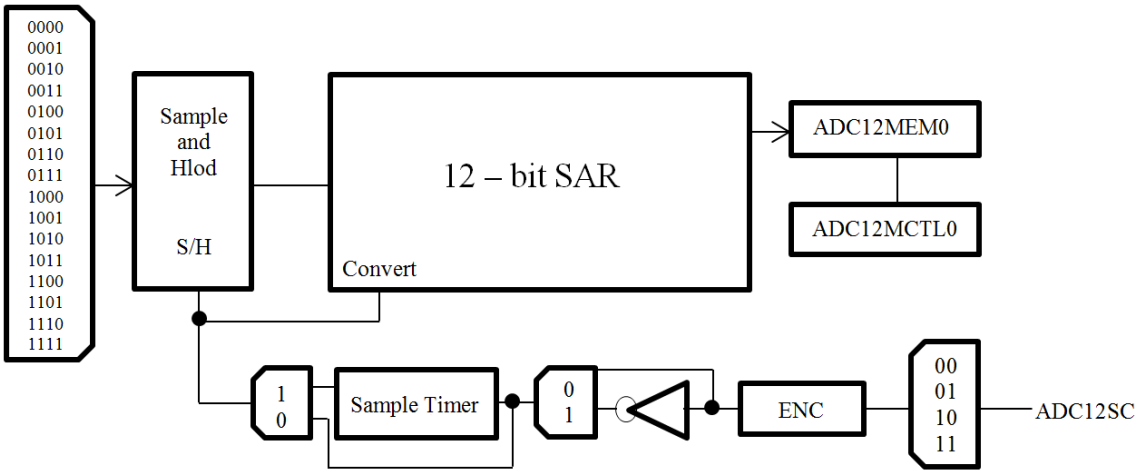


Obr. 3.1: Registr pro ADC12CTL0

Zapojení převodníku můžeme pozorovat na obr. 3.2 a přehled jeho funkcí v ukázce kódu 3.2 .

1. Nejprve nastavíme analog-digitální registr a příslušnou paměť pro zpracování, pak zapneme převodník a určíme držení 1 a 0 s uložením v dané paměti.
2. Nastavíme a inicializujeme skokový režim pro uložení 1 a 0.
3. Povolíme přerušení pro určitý kanál.
4. V daném registru povolíme konverzi.

5. Vybereme vstupní kanál pro konverzi.
6. Zvolíme výstupní kanál.



Obr. 3.2: Zapojení AD převodníku

Ukázka kódu 3.2: Analog-digital převodník

```
ADC12CTL0 = ADC12ON + ADC12SHT02;
ADC12CTL1 = ADC12SHP;
ADC12IE = 0x01;
ADC12CTL0 |= ADC12ENC;
P6SEL |= 0x01;
P1DIR |= 0x01;
```

V ukázce kódu 3.3 vytvoříme soubor, do kterých se nám bude ukládat posloupnost náhodných generovaných čísel. Můžeme jej vytvořit s binární příponou, či textovou. Pokud generujeme více jak jednou do stejného souboru, tak dojde ke smazání předešlých dat.

Ukázka kódu 3.3: Vytvoření souboru

```
char noise[15001];
fp=fopen("nahodne_cisla.txt", "w");
```

Nastavíme pole o určité velikosti, do kterého se po sepnutí převodníku začnou ukládat data. Hodnoty jsou ukládány do paměti ADC12MEM0. Dále nastavíme nízkovýkonový mód LPM0 pro mikrokontroler.

Ukázka kódu 3.4: Inicializace převodníku

```
while (j <= 15001) {
ADC12CTL0 |= ADC12SC;
  i = ADC12MEM0 % 2;
  __bis_SR_register(LPM0_bits + GIE);
  __no_operation();
```

Uloženým hodnotám (již 1 a 0) pomocí podmínek přesně definujeme zápis, jakým se mají ukládat do souboru. Následně jsou hodnoty příkazem fwrite ukládány do již vytvořeného prázdného pole v souboru. Jelikož se nám čísla generují pouze v poli o rozsahu 15 000 znaků, musíme doprogramovat smyčku, která se nám bude opakovat, pokud podmínka dosáhne požadovaného počtu uložených dat.

Ukázka kódu 3.5: Inicializace generování

```
if (j >= 15000) {
for(k = 0; k < 70; k++) {
  fwrite(noise, sizeof(char), 15000*sizeof(char), fp);
}
fclose(fp);
```

Poslední část programu je funkce sloužící pro funkčnost programu, kde se za pomoci blikání diody ověřuje chod programu. A nakonec vypnutí procesoru.

Ukázka kódu 3.6: Pragma vektor

```
#pragma vector = ADC12_VECTOR
__interrupt void ADC12_ISR(void)
{
if (ADC12MEM0 >= 0x7FF) {
P1OUT |= BIT0;
}
else
{
 P1OUT &= ~BIT0;
}
 _BIC_SR_IRQ(CPUOFF);
}
```

3.3 Ověření náhodnosti vlastního návrhu generátoru

Kvalitně provedený pravý generátor náhodných čísel by měl projít všemi druhy testů NIST. Vstupní data pro provedení testů mohou být dodány jako samostatný soubor čísel, nebo můžeme naimplementovat vlastní generátor do výchozího programu NIST. Pro provedení testů by dodávané soubory měli obsahovat binární sekvenci 0 a 1 uloženou jako ASCII znaky, nebo jako hexadecimální znaky uložené v binárním formátu. Testy vlastního implementovaného generátoru náhodných čísel byly provedeny na vstupních datech ze souboru, které byly generovány hardwarovým generátorem za pomocí šumu. Vstupní testovaná data byly voleny délky 1 000 000 bitů, pro kritickou hodnotu $\alpha = 0{,}01$, pod kterou nesmí výsledné P-hodnoty klesnout, jinak

je test zamítnut alternativní hypotézou H_A. Výsledky otestované sekvence vygenerované vlastním návrhem generátoru náhodných čísel za pomoci baterie testů NIST můžeme pozorovat v tabulce 3.1. Výsledné hodnoty testů splňují testovací hypotézu H_0, podle podmínky $P - hodnota \geq \alpha$, kromě testu náhodných exkurzí a náhodných exkurzních variant, které neuspěli z důvodu špatného rozložení nul v testované posloupnosti.

Tab. 3.1: Výsledky statistických testů NIST vlastního návrhu generátoru

Typ testu	**Výsledek testu**
Frekvenční test	0,16758
Frekvenční test v rámci bloku	0,36636
Test série bitů	0,49702
Test na nejdelší běh jedniček v bloku	0,15001
Test sérií binárních matic	0,15749
Test Diskrétní Fourierovy transformace	0,05396
Test nepřekrývajících se vzorů	0,57245
Test překrývajících se vzorů	0,58518
Mauerův univerzální statický test	0,58518
Test lineární složitosti	0,74382
Test serií	0,87422
Test entropie	0,07107
Test kumulačních součtů	0,27773

3.3.1 Srovnání s jinými výsledky

Pokud bych měl srovnat výsledky s jinými generátory, například od HotBits, který testoval posloupnost větší jak 16 000 000 bitů, nebo od Raiden, který testoval posloupnost 2 000 000 bitů (výsledky můžete vidět v tabulce 3.2 [21] [22]), tak lze považovat výsledky výše uvedených statistických testů NIST za průměrné. Nevýhodou fyzikálních a chemických generátorů je, že generované sekvence nelze předvídat, proto nelze zaručit, že testovaná sekvence čísel bude vždy hodnocena jako úspěšná.

Tab. 3.2: Výsledné porovnání testů od HotBits a Raiden s vlastními výsledky testů NIST

Typ testu	HotBits	Raiden	NIST
Frekvenční test	0.09093	0,602	0,16758
Frekvenční test v rámci bloku	0.27570	0,862	0,36636
Test série bitů	0.23276	0,213	0,49702
Test na nejdelší běh jedniček v bloku	0.00056	0,066	0,15001
Test sérií binárních matic	0.40709	0,178	0,15749
Test Diskrétní Fourierovy transformace	0.68901	0,739	0,05396
Test nepřekrývajících se vzorů	0.58520	0,178	0,57245
Test překrývajících se vzorů	0.65446	0,009	0,58518
Mauerův univerzální statický test	0.51744	0,101	0,58518
Test lineární složitosti	0.67177	0,101	0,74382
Test serií	0.67177	0,124	0,87422
Test entropie	0.75647	0,862	0,07107
Test kumulačních součtů	0.09561	0,303	0,27773

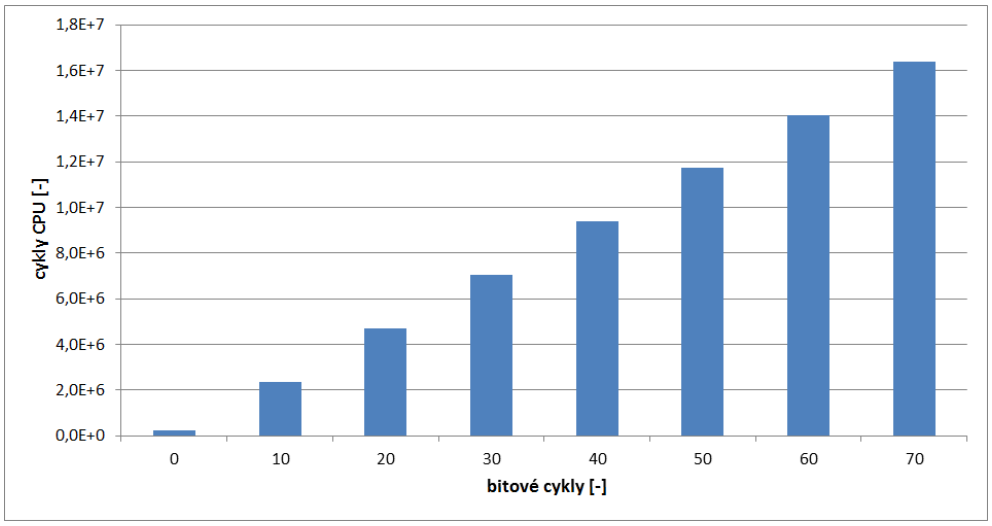
3.4 Optimalizace programu

Optimalizací programu, je myšlen proces modifikace výpočetního systému, který vede k vyšší efektivitě a ke snížení nároků celého výpočetního systému.
Program optimalizujeme z hlediska:

- rychlejšího spuštění,
- rychlejšího zpracování,
- menší náročnosti operační paměti,
- méně systémových prostředků.

Optimalizovat program můžeme z hlediska zdrojového kódu, ale i pomocí registrů či cyklů. V mé práci se zabývám optimalizací rychlosti kompilace a alokace registrů.
Model procesorové náročnosti generování neoptimalizovaného programu můžeme vidět na obr. 3.3. Jeden bitový cyklus generuje 15 000 bitů, kde pro každou takto vygenerovanou délku posloupnosti čísel je zapotřebí průměrně 236 503 cyklů centrální procesorové jednotky. Z grafu je patrné, že procesorová náročnost generování několika posloupností po sobě jdoucích je lineární, což je uzpůsobeno zdrojovým kódem, aby se náročnost operací s větším počtem generovaných čísel nezvyšovala.

Celková generovaná posloupnost činí 1 000 000 bitů, kde z hlediska paměti pro samotný proces generování bylo využito v mikrokontroleru 4480 bajtů (z 41856 bajtů) FLASH paměti a 1774 bajtů (z 16384 bajtů) RAM paměti.



Obr. 3.3: Procesorová náročnost generování náhodných čísel

K optimalizaci programu může být využita i hardwarová násobička (pro rychlé násobení a dělení), avšak pro nás nevhodná, jelikož se spíše využívá u pseudonáhodných generátorů z hlediska rychlejší varianty výpočtu v algoritmu. Pro optimalizaci vlastního návrhu kódu bylo využito optimalizační úrovně kompilace kódu a optimalizace rychlosti.

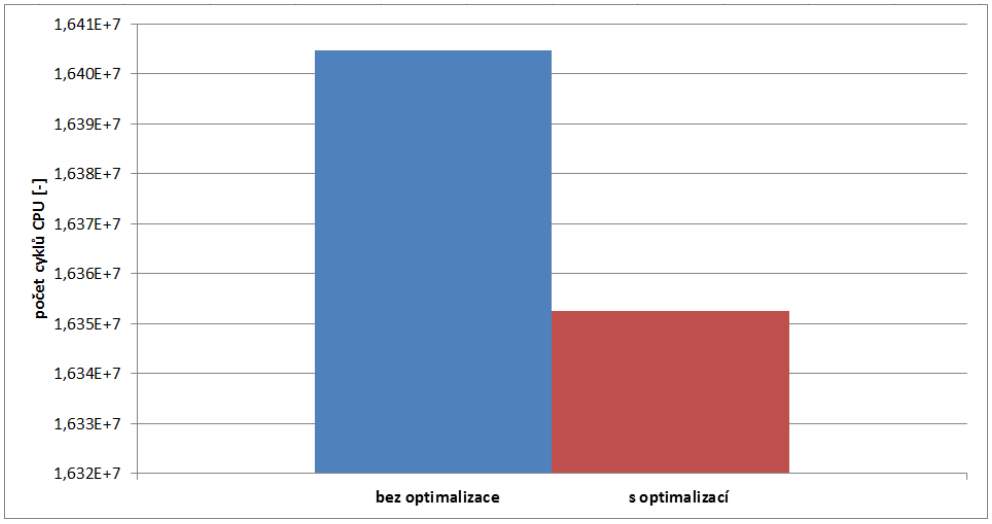
- Optimalizační úroveň kompilátoru – obsahuje 4 úrovně optimalizace, kde
  - úroveň 0 zahrnuje optimalizační registr,
  - úroveň 1 označuje lokální optimalizaci,
  - úroveň 2 přidává globální optimalizaci,
  - úroveň 3 obsahuje kompletní lokální i globální optimalizaci,
  - úroveň 4 zahrnuje do optimalizace celý program.
- Optimalizace rychlosti – obsahuje 5 úrovní, kde úroveň 5 označuje plnou optimalizaci pro rychlost ve smyslu zlepšení výkonu, avšak minimální optimalizaci určenou pro kompilaci velikosti kódu. S úrovní 0 je o přesně naopak.

Z hlediska kvalitní optimalizace se doporučují úrovně mezi horní a dolní hranicí, nejlépe úroveň 3. S využitím softwarové optimalizace vývojového prostředí Code Composer Studia, nastavenou na úroveň 3, kterou nám předurčila sama aplikace (viz výše), je možné minimalizovat náročnost procesorové jednotky až o 747 cyklů na jednu vygenerovanou posloupnost 15 000 bitů, výsledné srovnání výpočetní náročnosti CPU s optimalizací a bez optimalizace můžeme vidět v tab. 3.3. Pokud

generujeme posloupnost obsahující 1 000 000 bitů, s využitím výše zmíněné optimalizace, snížíme výpočetní náročnost až o 52 150 cyklů, tuto závislost můžeme pozorovat na obr. 3.4. Po následné optimalizaci velikosti kódu a rychlosti pro vygenerovanou posloupnost 1 000 000 bitů, bylo z hlediska paměti pro samotný proces generování využito v mikrokontroleru 4312 bajtů (z 41856 bajtů) FLASH paměti a 1774 bajtů (z 16384 bajtů) RAM paměti.

Tab. 3.3: Srovnání výpočetní náročnosti CPU s optimalizací a bez optimalizace

Optimalizace	**Cykly CPU**
Bez optimalizace	236 505
Velikosti kódu a rychlosti	235 758



Obr. 3.4: Výpočetní náročnost generování náhodných čísel optimalizovaného a neoptimalizovaného programu

4 ZÁVĚR

Cílem práce byl vlastní návrh hardwarového generátoru na bázi šumu s využitím nízko-výkonového zařízení.

Po úvodu obecné a informační bezpečnosti, která je počátkem inicializace těchto generátorů, byly nastíněny metody pro generování náhodných čísel, jež hrají důležitou roli v oblasti kryptografie, byť pro přímé šifrování náhodným heslem nebo pro generování privátních a veřejných klíčů. Byla provedena klasifikace metod generování za pomocí mechanických, fyzických či aritmetických generátorů, kde pro vlastní implementaci hardwarového generátoru byl zvolen fyzický generátor, založený na výstupní veličině určitého kanálu. Pro snadnou přenositelnost bylo využito nízkovýkonového mikrokontroleru MSP430.

V poslední části práce je za pomocí statistických testů NIST ověřena náhodnost posloupnosti vygenerovaných bitů pro vstupní data o velikosti 1 MB. Na základě testů a srovnáním s pseudo-generátory bylo vyhodnoceno, že využití hardwarového generátoru není příliš vhodné z hlediska rychlosti generování a přesného rozložení 0 a 1. Z hlediska efektivity by bylo vhodné využít smíšeného generátoru, spojením hardwarového se softwarovým generátorem by jsme získali výhody obou generátorů. Případná data zašifrovaná takovou posloupností skutečně náhodných čísel, by byla rovnoměrně rozložena.

LITERATURA

[1] PricewaterhouseCooper – informační bezpečnost [online]. c 2013-2015 [cit. 5. 5. 2015]. Dostupné na URL: <http://www.pwc.com/cz/cs/rizeni-rizik/informacni-bezpecnost.jhtml>.

[2] NOVÁK, L., POŽÁR, J. Systém řízení informační bezpečnosti. CyberSecurity.cz – Kybernetická bezpečnost [online]. [cit. 20. 11. 2014]. Dostupné na URL: <http://www.cybersecurity.cz/data/SRIB.pdf>.

[3] BURDA, K. *Bezpečnost informačních systému* Brno: Vysoké učení technické v Brně 2013, 152 s. ISBN 978-80-214-4890-2.

[4] HAAHR. Mads. Introduction to Randomness and Random Numbers [online]. c 1998-2015 [cit. 13. 4. 2015]. Dostupné na URL: <https://www.random.org/randomness/>.

[5] NIST Random Number Generation NIST Information Technology Laboratory [online]. 2014 [cit. 5. 5. 2015]. Dostupné na URL: <http://csrc.nist.gov/groups/ST/toolkit/rng/index.html>.

[6] JUN, B. a KOCHER, P. The Intel Random Number Generator [online]. 1999 [cit. 5. 5. 2015]. Dostupné na URL: <http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf>.

[7] WALKER, J. HotBits: Genuine Random Numbers [online]. c 1996-2006 [cit. 5. 5. 2015]. Dostupné na URL: <http://www.fourmilab.ch/hotbits/>.

[8] ZEMAN, V. Přednáška z předmětu Kryptografie v informatice: Generátory náhodných čísel [E-learning VUT Brno]. 2015 [cit. 13. 4. 2015]. Dostupné na URL: <https://www.vutbr.cz/elearning/file.php/149773/2015/MKRI_4_2015_prgn.pdf>.

[9] Eknihovna MU: Generování náhodných čísel [online]. [cit. 5. 5. 2015]. Dostupné na URL: <https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=7024>.

[10] GLOMBÍKOVÁ, V. Generování náhodných čísel : Pseudonáhodná čísla Přednáška [online]. 2008 [cit. 5. 5. 2015]. Dostupné na URL: <http://www.kod.tul.cz/info_predmety/Psi/prednasky_2007/prednaska_08/Generovan_08.pdf>.

[11] Environmental Protection Department – noise [online]. 2015 [cit. 5. 5. 2015]. Dostupné na URL: <http://www.epd.gov.hk/epd/noise_education/young/eng_young_html/m1/m1.html>.

[12] POSPÍŠIL, M. Šumový generátor [online]. 2008 [cit. 5. 5. 2015]. Dostupné na URL: <https://dspace.vutbr.cz/handle/11012/15208>.

[13] ZEMAN, V. Přednáška z předmětu Kryptografie v informatice: modulární aritmetika [E-learning VUT Brno]. 2015 [cit. 13. 4. 2015]. Dostupné na URL: <https://www.vutbr.cz/elearning/file.php/149773/2015/MKRI_1_2015.pdf>.

[14] BISKUP, R. Vyhodnocování výsledků testování hypotéz na základě „p-value“ [online]. [cit. 6. 5. 2015]. Dostupné na URL: <http://home.ef.jcu.cz/~birom/stat/cviceni/09/p_value.pdf>.

[15] NIST Special Publication 800-22 [online]. 2001 [cit. 5. 5. 2015]. Dostupné na URL: <http://csrc.nist.gov/groups/ST/toolkit/rng/index.html>.

[16] MARSAGLIA, G. DIEHARD Statistical Tests [online]. 1995 [cit. 5. 5. 2015]. Dostupné na URL: <http://stat.fsu.edu/~geo/diehard.html>.

[17] A Pseudorandom Number Sequence Test Program – ENT [online]. [cit. 5. 5. 2015]. Dostupné na URL: <http://www.fourmilab.ch/random/>.

[18] SOTO, J. National Institute of Standards and Technology: Statistical Testing of Random Number Generators [online]. [cit. 5. 5. 2015]. Dostupné na URL: <http://csrc.nist.gov/groups/ST/toolkit/rng/documents/nissc-paper.pdf>.

[19] User's Guide MSP430x5xx and MSP430x6xx Family [online]. 2014 [cit. 5. 5. 2015]. Dostupné na URL: <http://www.ti.com/lit/ug/slau208n/slau208n.pdf>.

[20] Mixed signal microcontroller – TI MSP430 [online]. 2015 [cit. 5. 5. 2015]. Dostupné na URL: <http://en.wikipedia.org/wiki/TI_MSP430>.

[21] WALKER, J. HotBits Statistical Testing [online]. 2006 [cit. 5. 5. 2015]. Dostupné na URL: <https://www.fourmilab.ch/hotbits/statistical_testing/stattest.html>.

[22] Raiden – Results of NIST Battery [online]. [cit. 5. 5. 2015]. Dostupné na URL: <http://raiden-cipher.sourceforge.net/statistical_tests.html>.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ADC Analogově číslicový převodník – Analog to Digital Converter

ASCII Americký standardní kód pro výměnu informací – American Standard Code for Information Interchange

CCS Code Composer Studio

FRAM Feroelektrická paměť – Ferroelectric Random Access Memory

ICT Informační a komunikační technologie – Information and Communication Technologies

LPM Počet řádek za minutu – Lines Per Minute

LSB Nejméně významný (řádově) bit – Least Significant Bit

MCU Řídící jednotka stroje – Machine Control Unit

MSP Skupina autorských týmů vyvíjejících projekty a řešení na bázi SW produktů Microsoft – Microsoft Solution Provider

NIST Národní institut standardů a technologie – National Institute of Standards and Technology

PRNG Softwarový generátor náhodných čísel – Pseudo-Random Number Generator

RAM Paměť s libovoným přístupem – Random Access Memory

TRNG Pravý „hardwarový“ generátor náhodných čísel – True Random Number Generator

USB Univerzální sériová sběrnice – Universal Serial Bus

SEZNAM PŘÍLOH

A OBSAH PŘILOŽENÉHO CD

- **HW generátor**
  - vlastní nárvh hardwarového generátoru náhodných čísel s využitím mikrokontroleru MSP430. Software byl testován ve verzi Code Composer Studio verze 5.2.1.00018
- **xecler01.pdf**
  - text bakalářské práce